23 | 实战：如何用深度学习模型实现Sparrow RecSys的个性化推荐功能?

2020-12-04 王喆

深度学习推荐系统实战 进入课程 >

讲述：王喆
时长 14:22 大小 13.16M

你好，我是王喆。

今天又是一堂实战课。在这节课里，我会带你利用我们现阶段掌握的所有知识，来实现SparrowRecSys 中“猜你喜欢”的功能。具体来说，我们会根据一位用户的历史行为，为TA 推荐可能喜欢的电影。这个功能几乎会用到所有的推荐系统模块，包括离线的特征工程、模型训练以及线上的模型服务和推荐逻辑的实现。

如果说完成了第 14 节课的“相似电影”功能，还只是你“武功小成”的标志，那啃下这节课的实践，就代表你掌握了推荐系统技术框架中的大部分内容，你就能在推荐系统的实际工作中做到“驾轻就熟”啦。

“清点技能库”，看看我们已有的知识储备有哪些

正式开始实践之前，我们还是先来清点一次自己的技能库。看看经过推荐模型篇的学习，我们技能库中的“兵器”又增加了多少，哪些可以用来实现“猜你喜欢”这个功能。下面，我就按照从离线到线上，由数据到模型的顺序，为你依次梳理一下特征工程、模型离线训练、模型服务、推荐服务器逻辑这四大部分的技能点。

1. 模型特征工程

特征工程是所有机器学习项目的起点，咱们的推荐模型也不例外。为了训练推荐模型，我们需要准备好模型所需的样本和特征。此外，在进行模型线上推断的时候，推荐服务器也需要线上实时拼装好包含了用户特征、物品特征、场景特征的特征向量，发送给推荐模型进行实时推断。

在“模型实战准备二”这节课，我们就通过 Spark 处理好了 TensorFlow 训练所需的训练样本，并把 Spark 处理好的特征插入了 Redis 特征数据库，供线上推断使用。不熟悉这部分内容的同学，最好再复习一下相关内容，把这把武器装进自己的技能库。

2. 模型离线训练

为了在线上做出尽量准确或者说推荐效果尽量好的排序，我们需要在离线训练好排序所用的推荐模型。

我们在这一篇中学习和实践的所有深度推荐模型，都是围绕着这个目的展开的。虽然这些深度推荐模型的结构各不相同，但它们的输入、输出都是一致的，输入是由不同特征组成的特征向量，输出是一个分数，这个分数的高低代表了这组特征对应的用户对物品的喜好程度。

具体实践的时候，我们在 TensorFlow 平台上实现了 Embedding MLP、Wide&Deep、NeuralCF、双塔模型、DeepFM 等几种不同的深度推荐模型，它们中的任何一个都可以支持“猜你喜欢”的排序功能。

在实际的工业级系统中，我们会通过离线、在线等不同的评估手段来挑出最好的那个模型，去支持真实的应用场景。在 SparrowRecsys 中，我们以 NeuralCF 模型为例，实现“猜你喜欢”功能。其他模型的上线方法与 NeuralCF 几乎一致，唯一的区别是，对于

不同的模型来说，它们在模型服务的部分需要载入不同的模型文件，并且在线上预估的部分也要传入模型相应的输入特征。

3. 模型服务

模型服务是推荐系统中连接线上环境和线下环境的纽带之一（另一个关键的纽带是特征数据库）。

在离线训练好模型之后，为了让模型在线上发挥作用，做出实时的推荐排序，我们需要通过模型服务的模块把推荐模型部署上线。我们曾经在第 13 节课中详细介绍过主流的模型服务方法，它们是“预存推荐结果”，“预训练 Embedding+ 轻量级线上模型”，“利用 PMML 转换和部署模型”以及“TensorFlow Serving”。因为我们这一篇的深度学习模型都是基于 TensorFlow 训练的，所以这节课我们也会采用 TensorFlow Serving 作为模型服务的方式。

4. 推荐服务器内部逻辑实现

模型服务虽然可以做到“猜你喜欢”中电影的排序，但要进行排序，仍然需要做大量的准备工作，比如候选集的获取，召回层的构建，特征的获取和拼装等等。这些推荐逻辑都是在推荐服务器内部实现的。推荐服务器就像推荐系统的线上的心脏，是所有线上模块的核心。

我们曾经在“相似电影”功能中实现过整套的推荐逻辑，今天我们重点关注其中不同的部分，就是特征的拼装，以及从推荐服务器内部请求模型服务 API 的方法。

至此，我们准备好了自己的技能库。接下来，就让我们使出十八般武艺，来打造“猜你喜欢”这个推荐功能吧。

“猜你喜欢”推荐功能的技术架构

与“相似电影”功能一样，“猜你喜欢”相关的技术架构同样是由数据模型部分、线上部分和前端部分组成的。我们先来看看整个功能的技术架构图，再来说说每部分的具体实现细节。下图 1 就是“猜你喜欢”功能的技术架构图，接下来，你就跟着我，按照从左上到右下的顺序，一起随着数据流的走向过一遍这个架构图吧。

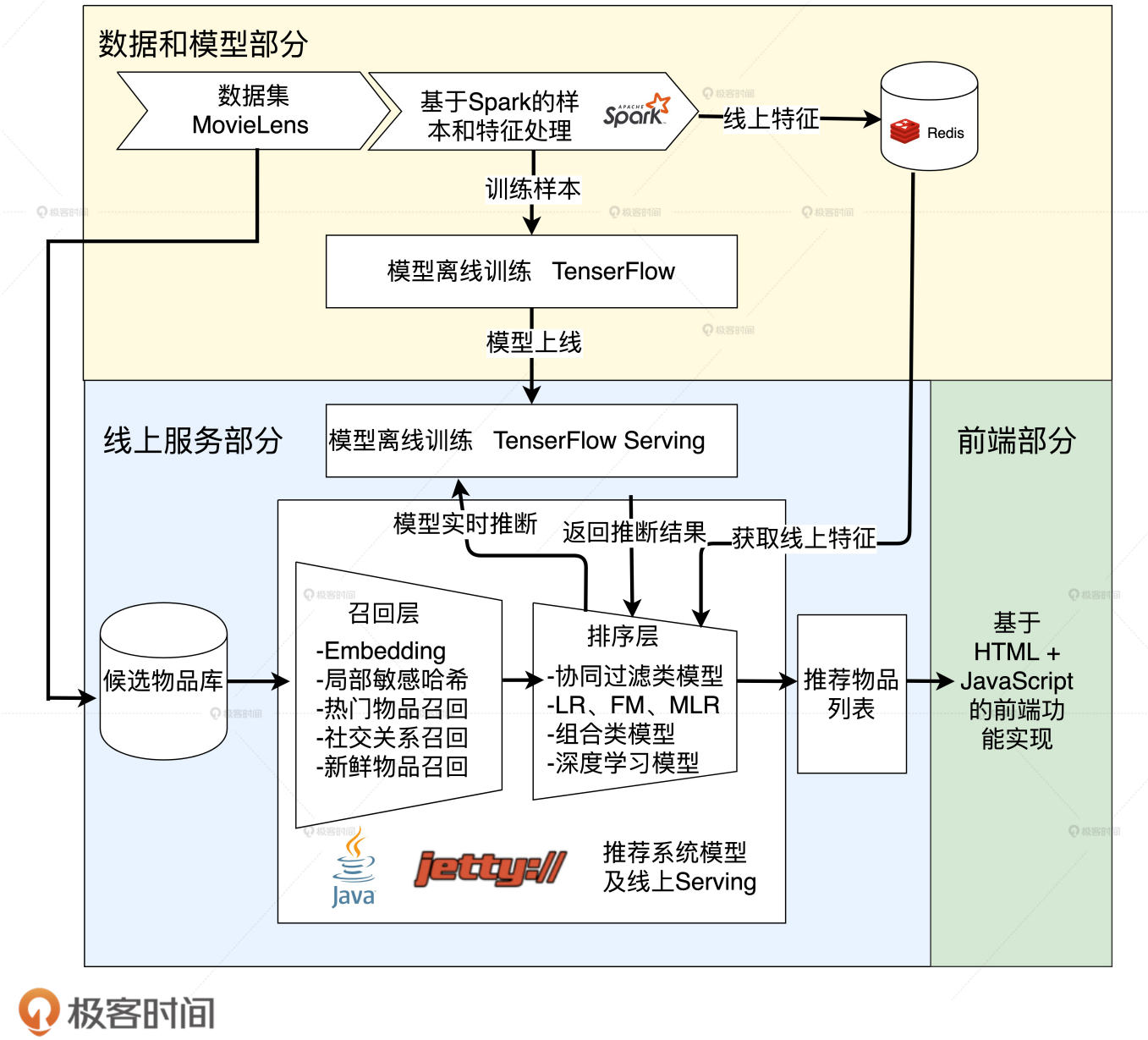


图1 “猜你喜欢”功能的技术架构图

首先，我们来看数据和模型部分。左上角是我们使用的数据集 MovieLens，它经过 Spark 的处理之后，会生成两部分数据，分别从两个出口出去，特征部分会存入 Redis 供线上推断时推荐服务器使用，样本部分则提供给 TensorFlow 训练模型。

TensorFlow 完成模型训练之后，会导出模型文件，然后模型文件会载入到 TensorFlow Serving 中，接着 TensorFlow Serving 会对外开放模型服务 API，供推荐服务器调用。

接下来，我们再看推荐服务器部分。在这部分里，基于 MovieLens 数据集生成的候选电影集合会依次经过候选物品获取、召回层、排序层这三步，最终生成“猜你喜欢”的电影推荐列表，然后返回给前端，前端利用 HTML 和 JavaScript 把它们展示给用户。

整个过程中，除了排序层和 TensorFlow Serving 的实现，其他部分我们都已经在之前的实战中一一实现过。所以今天，我们会重点讲解推荐服务器排序层和 TensorFlow Serving 的实现。

排序层 +TensorFlow Serving 的实现

在推荐服务器内部，经过召回层之后，我们会得到几百量级的候选物品集。最后我们到底从这几百部电影中推荐哪些给用户，这个工作就交由排序层来处理。因为排序的工作是整个推荐系统提高效果的重中之重，在业界的实际应用中，往往交由评估效果最好的深度推荐模型来处理。整个的排序过程可以分为三个部分：

1. 准备线上推断所需的特征，拼接成 JSON 格式的特征样本；
2. 把所有候选物品的特征样本批量发送给 TensorFlow Serving API；
3. 根据 TensorFlow Serving API 返回的推断得分进行排序，生成推荐列表。

接下来，我们就详细来讲讲这三步中的实现重点。

首先，第一步的实现重点在于特征样本的拼接。因为实践例子里，我们选用了 NeuralCF 作为排序模型，而 NerualCF 所需的特征只有 `userId` 和 `itemId` ，所以特征是比较好准备的。我们下面看一下如何拼接特征形成模型推断所需的样本。详细的代码，你可以参考 com.wzhe.sparrowrecsys.online.recprocess.RecForYouProcess。

复制代码

```
/**
 * call TenserFlow serving to get the NeuralCF model inference result
 * @param user              input user
 * @param candidates        candidate movies
 * @param candidateScoreMap save prediction score into the score map
 */
public static void callNeuralCFTFServing(User user, List<Movie> candidates, Ha
    if (null == user || null == candidates || candidates.size() == 0){
        return;
    }
    //保存所有样本的JSON数组
    JSONArray instances = new JSONArray();
    for (Movie m : candidates){
        JSONObject instance = new JSONObject();
        //为每个样本添加特征，userId和movieId
        instance.put("userId", user.getUserId());
        instance.put("movieId", m.getMovieId());
```

```
18          instances.put(instance);
19      }
20      JSONObject instancesRoot = new JSONObject();
21      instancesRoot.put("instances", instances);
22      //请求TensorFlow Serving API
23      String predictionScores = asyncSinglePostRequest("http://localhost:8501/v1
24      //获取返回预估值
25      JSONObject predictionsObject = new JSONObject(predictionScores);
26      JSONArray scores = predictionsObject.getJSONArray("predictions");
27      //将预估值加入返回的map
28      for (int i = 0 ; i < candidates.size(); i++){
29          candidateScoreMap.put(candidates.get(i), scores.getJSONArray(i).getDou
30      }
31  }
```

在代码中，我们先把 userId 和 movieId 加入了 JSON 格式的样本中，然后再把样本加入到 Json 数组中。接下来，我们又以 http post 请求的形式把这些 JSON 样本发送给 TensorFlow Serving 的 API，进行批量预估。在收到预估得分后，保存在候选集 map 中，供排序层进行排序。

第二步的重点在于如何建立起 TensorFlow Serving API。事实上，我们通过第 13 讲模型服务的实践部分，已经能够搭建起一个测试模型的 API 了。

想要搭建起我们自己的 TensorFlow Serving API，只需要把之前载入的测试模型文件换成我们自己的模型文件就可以了。这里，我就以 NerualCF 模型为例，带你看一看模型文件是如何被导出和导入的。

首先是模型的导出。在 NeuralCF 的 TensorFlow 实现中，我们已经把训练好的模型保存在了 model 这个结构中，接下来需要调用 tf.keras.models.save_model 这一函数来把模型序列化。

从下面的代码中你可以看到，这一函数需要传入的参数有要保存的 model 结构，保存的路径，还有是否覆盖路径 overwrite 等等。其中，我们要注意的是保存路径。你可以看到，我在保存路径中加上了一个模型版本号 002，这对于 TensorFlow Serving 是很重要的，因为 TensorFlow Serving 总是会找到版本号最大的模型文件进行载入，这样做，就保证了我们每次载入的都是最新训练的模型。详细代码请你参考 NeuralCF.py。

复制代码

```
1
```

```
tf.keras.models.save_model(
    model,
  "file:///Users/zhewang/Workspace/SparrowRecSys/src/main/resources/webroot/mo
    overwrite=True,
    include_optimizer=True,
    save_format=None,
    signatures=None,
    options=None
)
```

其次是模型的导入，导入命令非常简单就是 TensorFlow Serving API 的启动命令，我们直接看下面命令中的参数。

复制代码

```
docker run -t --rm -p 8501:8501     -v "/Users/zhewang/Workspace/SparrowRecSys
```

这里面最重要的参数，就是指定载入模型的路径和预估 url，而载入路径就是我们刚才保存模型的路径，“/Users/zhewang/Workspace/SparrowRecSys/src/main/resources/webroot/modeldata/neuralcf”。但是在这里，我们没有加模型的版本号。这是为什么呢？因为版本号是供 TensorFlow Serving 查找最新模型用的，TensorFlow Serving 在模型路径上会自动找到版本号最大的模型载入，因此不可以在载入路径上再加上版本号。

除此之外，冒号后的部分“/models/recmodel”指的是 TensorFlow Serving API 在这个模型上的具体 url，刚才我们是通过请求 http://localhost:8501/v1/models/recmodel:predict 获取模型预估值的，请求连接中的 models/recmodel 就是在这里设定的。

在正确执行上面的命令后，我们就可以在 Docker 上运行起 TensorFlow Serving 的 API 了。

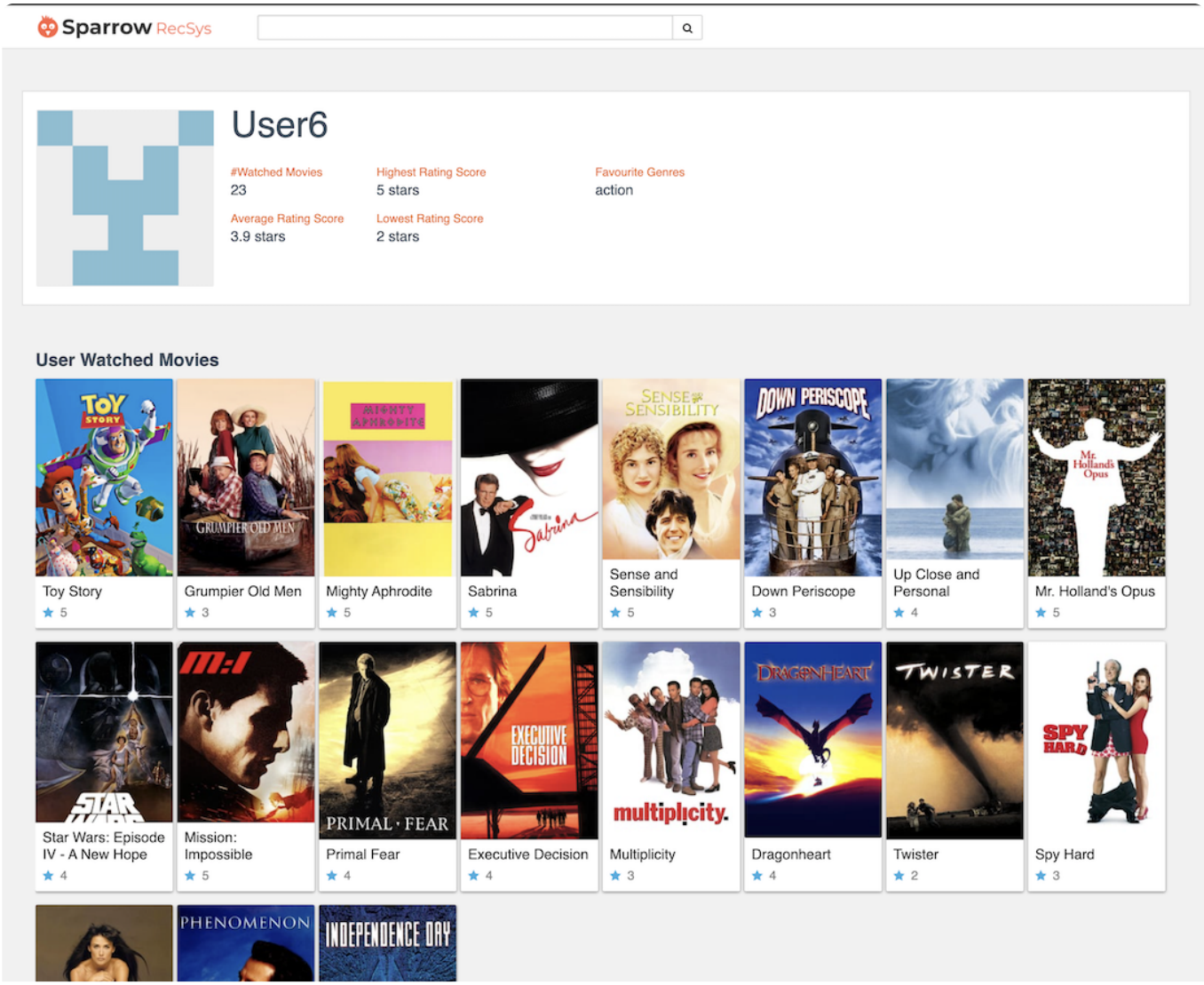
最后，我们来看第三步的实现重点：获取返回得分和排序。 我们先来看一下 TensorFlow Serving API 的返回得分格式。它的返回值也是一个 JSON 数组的格式，数组中每一项对应着之前发送过去的候选电影样本，所以我们只要把返回的预估值赋给相应的样本，然后按照预估值排序就可以了。详细的过程你也可以参考 com.wzhe.sparrowrecsys.online.recprocess.RecForYouProcess 中全部排序层的代码。

复制代码

```
{
    "predictions": [[0.824034274], [0.86393261], [0.921346784], [0.957705915],
    ]
}
```

如果你已经正确建立起了 Redis 和 TensorFlow Serving API 服务，并且已经分别导入了特征数据和模型文件，我们就可以启动 SparrowRecsys Server，查看“猜你喜欢”的结果了。图 2 是用户 ID 为 6 的用户在 NerualCF 模型下的推荐结果，注意通过在连接中设置 model 变量为 nerualcf，来决定产生结果的模型。

通过用户的评分历史（User Watched Movies）我们可以看到该用户偏向于观看动作类的电影，同时夹杂着一些爱情片和动画片，而在下方的“猜你喜欢”（Recommended For You）的结果中，我们也可以看到 SparrowRecsys 为他推荐的电影也包含了这三类电影。有兴趣的话，你可以多在 SparrowRecsys 里面逛一逛，看看推荐的结果在你眼中是不是合理。



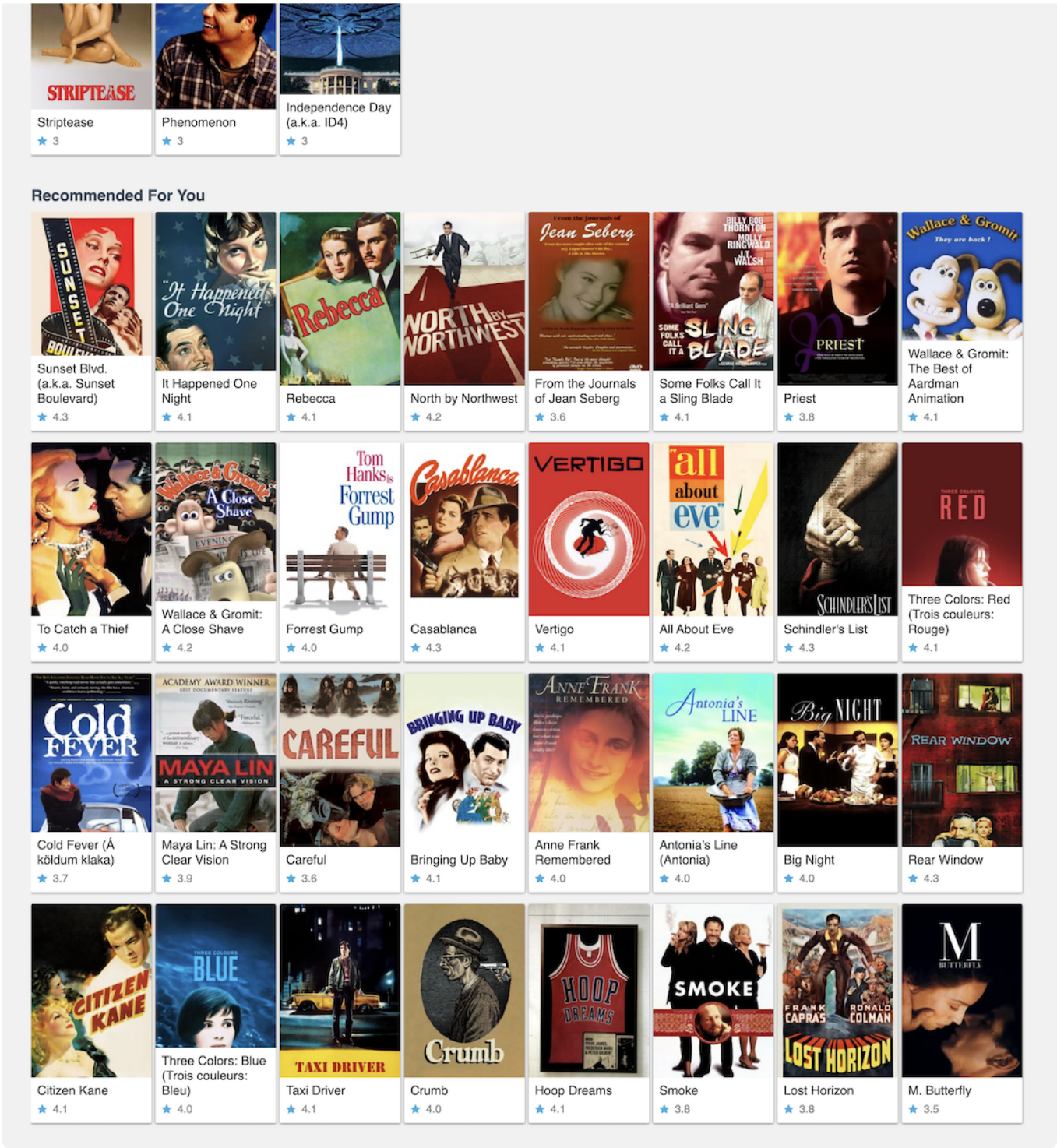


图2 猜你喜欢功能的推荐结果

小结

今天我们通过实现“猜你喜欢”功能串联起了我们之前所有学过的知识。希望在你看到推荐结果的时候，有种“武功大成，驾轻就熟”的感觉。要知道，这里面所有的代码都是你曾经学习过的，这里面每个结果都是你通过自己的所学生成的。希望你能在这里为自己鼓掌，这是一个不小的里程碑。

下面我们再重点总结一下今天实践用到的技术。首先，我们利用 Spark 对 MovieLens 原始数据进行了处理，生成了训练样本和特征，样本供 TensorFlow 进行模型训练，特征存

入 Redis 供线上推断使用。

在 TensorFlow 平台上，我们以 NeuralCF 模型为例，训练并导出了 NeuralCF 的模型文件。然后使用 TensorFlow Serving 载入模型文件，建立线上模型服务 API。推荐服务器的排序层从 Redis 中取出用户特征和物品特征，组装好 JSON 格式的特征数据，发送给 TensorFlow Serving API，再根据返回的预估分数进行排序，最终生成“猜你喜欢”的推荐列表。

虽然我们实现了猜你喜欢的功能，但是课程进行到这里你一定会有一个疑问：我们的推荐结果到底是好还是坏呢？我们总不能总是人肉去查看结果好坏吧，这样效率又低，又不准确。没错，推荐系统的效果评估是有一套非常完整的评估体系的，别着急，从下一篇的模型评估篇开始，我们就会系统性地讲解推荐系统的评估方法，期待继续与你同行。

课后思考

推荐系统的特征预处理是一项很重要的工作，比如一些连续特征的归一化，分桶等等。那么这些预处理的过程，我们应该放在线上部分的哪里完成呢？是在 Tensorflow Serving 的部分，还是在推荐服务器内部，还是在离线部分完成？你有什么好的想法吗？

期待在留言区看到你的想法和思考，我们下一节课见！

提建议

更多学习推荐

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 特别加餐 | “银弹”不可靠，最优的模型结构该怎么找？

精选留言 (3)

写留言

Sebastian
2020-12-04

思考题：特征分为静态特征和动态特征。对于静态特征，基本长时间不会变更，这块直接从特征池（可以是Redis）里取。但是对于动态特征，比如用户实时行为的特征，这种会通过流式处理（比如spark streaming或者flink）后，直接落盘，同时可以避免特征穿越。特征实时更新后，线上服务阶段，模型的输入就是未进行处理的原格式数据，tf serving接受请求后，在模型里进行特征预处理，比如使用tf.feature_column进行处理，转为on...
展开

作者回复: 大致流程确实是这样的。但是其实在实际应用中，tf serving的延迟问题会比较严重。我们会想尽一切办法去减少tf serving的负担。所以特征预处理这块会尽量放到推荐服务器内部来做，或者放到离线做好预存到线上存储。

比如id 2 embeding这一步，可以把id和emb的对应关系放到redis里或者其他线上数据库，减轻tf serving的压力和模型体积。

Berton
2020-12-04

特征处理这部分，应该是离线计算好得到每个特征的map数据，在推荐服务器内部加载这些map数据，直接将原始特征映射成深度学习需要的向量，将得到的向量送入Tensorflow Serving得到推荐结果
如果在Tensorflow Serving做特征预处理的工作，会导致推荐服务的响应时间边长

展开

作者回复: 是的，最好是可以在推荐服务器内部把特征都准备好，处理好。tf serving只做inference，不承担太多特征预处理压力。

X.G
2020-12-04

基础架构篇里说，最终的系统会有这个功能：随着当前用户不断为电影打分，系统会对首页的推荐结果进行个性化的调整，比如电影类型的排名会进行个性化调整，每个类型内部的影片也会进行个性化推荐。请问这个打分功能也是在后期课程中增加吗？

展开

作者回复: 这个功能因为需要比较多的前端工作，所以暂时不会实现了。我会在后续SparrowRecsys项目的迭代中再实现。本质上来说这个功能和猜你喜欢功能是一样的，区别在于要在server端动态记录用户的当前评价历史。

2